อนุชาติ ลาสนธิ

ทรัพยากรที่น่าใช้ ใน FoxPro 2.x

ดึงเอาความสัมพันธ์ของข้อมูลแบบต่าง ๆ ในฟอกซ์โปรออกมาใช้อย่างเต็มที่เพื่อการทำงานที่มีประสิทธิภาพ

DBMS ที่มาแรงในขณะนี้คงปฏิเสธไม่ได้ว่าฟอกซ์โปรก็เป็นหนึ่งที่อยู่ในชั้นแนวหน้า (อันดับต้น ๆ ด้วย) จากรุ่นที่ 1.xx พัฒนามาเป็น 2.0 และล่าสุด Microsoft Foxpro 2.5 ซึ่งปัจจุบันก็มีการใช้งานกันกว้างขวางพอสมควร ผู้เขียนจึงมีเรื่องน่ารู้เกี่ยวกับการใช้งานมาแนะนำให้ท่านผู้อ่านได้รับรู้เรื่องราวของ Foxpro 2.x ซึ่งเรื่องที่จะเล่ามีอยู่ 6 เรื่องดังนี้ คือ

1. ความสัมพันธ์แบบ One to many
2. แฟ้ม index แบบคีย์รวม
3. การ index ด้วย UDFs
4. BROWSE
5. Text editor ของ FoxPro
6. Startup switches

ความสัมพันธ์แบบ One to many

ในการทำความเข้าใจความสัมพันธ์แบบ One to many ผู้เขียนขอใช้แฟ้มข้อมูล 3 แฟ้มประกอบการอธิบาย คือ name.dbf, company.dbf และ phon.dbf มีข้อมูลดังรูปที่ 1, 2 และ 3

การสร้างความสัมพันธ์ของแฟ้มข้อมูลไม่ว่าจะเป็นแบบ one to one หรือ one to many จะต้องมีแฟ้มข้อมูลหลักหนึ่งแฟ้มเรียกว่า parent database และแฟ้มข้อมูลที่จะมาสัมพันธ์ด้วยเรียกว่า child database เมื่อสร้างความสัมพันธ์กันด้วยคำสั่ง SET RELATION แล้ว ทุกครั้งที่เรคอร์ดพอยน์เตอร์ของ parent database เปลี่ยนไป เรคอร์ดพอยน์เตอร์ของ child database ก็เปลี่ยนตามไปด้วย โดยจะชี้ที่ matching record หากไม่มีจะชี้ที่ Eof( )

ขอย้อนกลับไปหาความสัมพันธ์แบบ one to one จากคำสั่งดังในรูปที่ 4 จะทำให้เกิดความสัมพันธ์แบบ one to one ของ 3 แฟ้มข้อมูล

name.dbf & id__name.idx

Record #	ID	NAME
4	1425	NIRAT
2	3421	RAWAT
1	5427	SOMCHAI
3	6473	ANURUK
5	7142	UDOM

รูปที่ 1 แฟ้มข้อมูล name.dbf & id__name.idx

company.dbf & id__compa.idx

Record #	ID	COMPANY
2	1425	QUASAR
7	1425	LEONICS
6	3421	PROCESSOR
4	5427	VALUE
1	6473	ACER
3	7142	MICROTEK
5	7142	SINAPIS

รูปที่ 2 แฟ้มข้อมูล company.dbf & id__compa.idx

มีการชี้ของเรคอร์ดพอยน์เตอร์ ดังตารางที่ 1

การสร้างความสัมพันธ์แบบ one to many จะใช้คำสั่ง set relation เพื่อสร้างแบบ one to one ก่อน จากนั้นจึงใช้คำสั่ง SET SKIP เพื่อให้เป็นแบบ one to many รูปแบบคำสั่งคือ

```
SET SKIP TO [<alias 1> [<alias 2>]...]
```

เมื่อสร้างความสัมพันธ์แบบ one to many แล้ว ทุกครั้งที่ใช้คำสั่ง skip ที่ parent database ตัวเรคอร์ดพอยน์เตอร์ของ parent database จะเลื่อนไปยังเรคอร์ดถัดไปหรือไม่ ขึ้นอยู่กับเมตชิงเรคอร์ดของ child database หากแมตชิงเรคอร์ดยังมีอยู่ คำสั่ง skip นี้ จะเป็นการเลื่อนเรคอร์ดพอยน์เตอร์ของ child database แทน จนกว่าจะไม่พบแมตชิงเรคอร์ด แล้วคำสั่ง skip จึงจะเป็นผลต่อ parent database

phon.dbf & id__phon.idx

Record #	ID	PHON
3	1425	3743026
2	3421	4364253
1	6473	4601655
4	6473	4253514
5	6473	3701452

รูปที่ 3 แฟ้มข้อมูล phon.dif & id__phon.idx

```
set status on
clear all
select 1
use name
index on id to id__name compact
select 2
use company
index on id to id__compa compact
select 3
use phon
index on id to id__phon compact
select name
set relation to id into company, id into phon
```

รูปที่ 4 คำสั่งที่ทำให้เกิดความสัมพันธ์แบบ one to one ของ 3 แฟ้มข้อมูล

คำสั่ง	name	company	phon
seek '1425'	4	2	3
skip	2	6	2
skip	1	4	eof( )
skip	3	1	1
skip	5	3	eof( )
skip	eof( )	eof( )	eof( )

ตารางที่ 1 การชี้ของเรคอร์ดพอยน์เตอร์ในความสัมพันธ์แบบ one to one

การสร้างความสัมพันธ์แบบ one to many ใช้คำสั่งดังในรูปที่ 5 และการชี้ของเรคอร์ดพอยน์เตอร์แสดงดังตารางที่ 2

รูปที่ 5
คำสั่งของการสร้าง
ความสัมพันธ์แบบ
one to many

```
clear all
select 1
use name index id__name
select 2
use company index id__compa
select 3
use phon index id__phon
select name
set relation to id into company, id into phon
set skip to company, phon
```

ตารางที่ 2
การชี้ของ
เรคอร์ดพอยน์เตอร์
ในความสัมพันธ์แบบ
one to many

คำสั่ง	name	company	phon
seek '1425'	4	2	3
skip	4	7	eof( )
skip	2	6	2
skip	1	4	eof( )
skip	3	1	1
skip	3	eof( )	4
skip	3	eof( )	5
skip	5	3	eof( )
skip	5	5	eof( )
skip	eof( )	eof( )	eof( )

แฟ้ม index แบบคีย์รวม

index file ที่เพิ่มเข้ามาใหม่ใน FoxPro 2.x มี 2 ชนิดคือ

- **Compact** เพียงแค่เพิ่มอนุประโยค "compact" ต่อท้ายเท่านั้นก็จะได้อินเด็กซ์ไฟล์แบบกระชับดังตัวอย่างที่ผ่านมา

- **Structural compound index** เป็นแฟ้มอินเด็กซ์ที่มีได้หลายคีย์ในแฟ้มเดียว จะถูกสร้างให้เป็นชนิดคอมแพคและมีนามสกุลเป็น .CDX อัตโนมัติ ยังไม่พอแค่นี้ structural index ยังแบ่งได้อีก 2 แบบ คือ

1. Structural index ที่มีชื่อเดียวกันกับชื่อแฟ้มข้อมูล และจะถูกเปิดใช้พร้อมกับการเปิดแฟ้มข้อมูลด้วยคำสั่ง USE (แต่จะยังไม่เรียงข้อมูลตามคีย์ใด ๆ จนกว่าจะสั่งด้วย SET ORDER) หากค้นหา .CDX ไม่พบ จะมีเออเรอเมสเสจแจ้งว่า "Structural CDX file not found" พร้อมให้ยืนยัน Ignore หรือ Cancel หากเลือก Ignore ฟอกซ์โปรจะตัดเครื่องหมายที่ใช้ระบุว่าแฟ้มนี้เป็นแบบ structural ออก พร้อมแสดงเมสเสจ "Structural CDX file reference removed" หมายความว่าแฟ้มข้อมูลนี้จะกลายเป็นแฟ้มธรรมดาทั่วไป ถ้าเลือก Cancel จะไม่มีการเปิดแฟ้มข้อมูล มีรูปแบบคำสั่งและตัวอย่างการใช้ ดังในรูปที่ 6

```
INDEX ON <expr> TAG <tag name>
[OF<.CDX file>]
```

Tag หรือออร์เดอร์หรือคีย์ที่จะใช้เรียงข้อมูลทันทีที่เปิดแฟ้มข้อมูลสามารถระบุได้พร้อมการเปิดแฟ้มข้อมูล เช่น

```
use name order tag name
```

หรือ

```
use name order 1
```

2. Structural index ที่มีชื่อไม่เหมือนกับแฟ้มข้อมูล อินเด็กซ์ชนิดนี้ใช้งานเหมือนกับคอมแพคอินเด็กซ์ทั่วไป (ไม่จำเป็นต้องเปิดพร้อมกับแฟ้มข้อมูล

```
clear all
use name
index on id tag id          * อินเด็กซ์ด้วย id ตั้งชื่อ tag name ว่า id
index on name tag name      * อินเด็กซ์ด้วย name ตั้งชื่อ tag name ว่า name
                            * ทั้งสองมี structural index file ชื่อ name.cdx
set order to tag id         หรือจะใช้ set order to 1 ก็ได้ * เลือก order 1
set order to tag name       หรือจะใช้ set order to 2 ก็ได้ * เลือก order 2
```

รูปที่ 6 ตัวอย่างการใช้คำสั่งเพื่อค้นหา .CDX

```
use name
index on id tag id of test          * อินเด็กซ์ด้วย id ตั้งชื่อ tag name
                                      ว่า id
index on name tag name of test      * อินเด็กซ์ด้วย name ตั้งชื่อ tag
                                      name ว่า name
```

รูปที่ 7 Structural index ที่มีชื่อไม่เหมือนกับแฟ้มข้อมูล

```
clear all
use name
set index to test                   * เปิดอินเด็กซ์ไฟล์
set order to 1                      * เลือกใช้ order 1 หรือ tag id
set order to tag name               * เลือกใช้ tag name หรือ order 2
use name index test order tag name  * เปิดแฟ้มข้อมูลพร้อมเลือกใช้ tag
                                      name, name
use name index test order 1         * เปิดแฟ้มข้อมูลพร้อมเลือกใช้
                                      order 1 หรือ tag id
```

รูปที่ 8 ตัวอย่างการเปิดใช้งาน Structural index ที่ชื่อ test.cdx

ก็ได้) แตกต่างกันตรงที่มีได้หลายคีย์กว่าเท่านั้น ดังในรูปที่ 7 จะได้ structural Index ชื่อ test.ccx และมี 2 คีย์ คือ id, name การเปิดใช้งานก็ทำเหมือนกับแฟ้มอินเด็กซ์ทั่วไป ดังในรูปที่ 8

Tag name ที่ไม่ต้องการสามารถลบออกจาก structural index ได้ด้วยคำสั่ง DELETE TAG ดังรูปแบบคำสั่ง

```
DELETE TAG <tag name1> [OF <.CDX file1>] [, <tag name2> [OF <.cdx file2]]...
```

หรือ

```
DELETE TAG ALL [OF <.CDX file 1>]
```

จากคำสั่งดังกล่าวจะได้ผลลัพธ์ดังรูปที่ 9

```
use name
index on id tag id
index on name tag name
delete tag id
delete tag all
use name
index on id tag id of test
index on name tag name of test
delete tag id of test
delete tag all of test
```

รูปที่ 9 ผลลัพธ์ที่ได้จากคำสั่ง DELETE TAG

มีเรื่องแปลกอยู่อย่างเกี่ยวกับคำสั่ง delete tag คือ tag name ใดที่ถูกลบแล้วจะเรียกกลับมาใช้ไม่ได้ (อันนี้ไม่แปลก) แต่ขนาดของ .CDX จะเท่าเดิม (อันนี้ซิแปลก)

index ด้วย UDFs (User Define Functions)

เรื่องนี้ผู้เขียนได้อ่านบทความของคุณประวิทย์ (ในไมโครฯ เล่ม 77) จึงทราบว่าคำสั่งอินเด็กซ์อนุญาตให้ใช้ UDFs ร่วมกับ <expr> ได้ จะทำให้การเรียงข้อมูลเป็นไปตามใจผู้เขียนว่างั้นเถอะ ตื้นเต้นใหญ่เลยถึงขนาดอ่านยังไม่จบก็มาลองทำดู ปรากฏว่า เอ๊ะ แล้วจะทำยังงัย? ทุกครั้งที่อินเด็กซ์ตัวฟอกซ์โปรจะไปเรียก UDFs เมื่อไหร่ และได้อย่างไร ต้องกลับไปนั่งคิดต่ออีกตั้งหลายวันกว่าจะคิดออก คืออย่างนี้ครับ เมื่อเราทำอินเด็กซ์โดยใช้ UDFs ทุก ๆ เรคอร์ดที่จะนำไปเรียงจะต้องผ่านค่ามายัง UDFs เสียก่อน คราวนี้ทุกสิ่งทุกอย่างก็จะเกิดขึ้นที่นี่ เราจะให้ความสำคัญอะไร อย่างไร ใครมาก่อนใครมาหลังก็ทำที่นี่ได้ทันที ผู้เขียนมีตัวอย่างการอินเด็กซ์ด้วย UDFs มาให้ดูหนึ่งฟังก์ชัน

โจทย์มีอยู่ว่า ให้เรียงข้อมูลด้วยคำสั่งอินเด็กซ์ โดยสนใจเฉพาะตัวอักษร 'A-2', 'a-2' เท่านั้น ตัวพิมพ์ใหญ่และตัวพิมพ์เล็กมีค่าเท่ากัน จากโจทย์จะได้ฟังก์ชันดังในรูปที่ 10 (ตั้งชื่อไฟล์เป็น ALPHA.PRG)

แฟ้มข้อมูลที่จะทดสอบโจทย์ดังกล่าวชื่อ udf_tst.dbf มีข้อมูลดังในรูปที่ 11

ขั้นตอนพิสูจน์

ดูในรูปที่ 12

หลังจากอินเด็กซ์ด้วยฟังก์ชัน

รูปที่ 10
ตัวอย่างฟังก์ชัน
ที่อินเด็กซ์ด้วย UDFS

```
1. FUNCTION alpha
2. PARAMETERS p__str
3.     dm = ''
4.     for i = 1 to len(p__str)
5.         ch = upper(subs(p__str,i,1))
6.         if between(ch,'A','Z') or between(ch,'a','z')
7.             dm = dm+ch
8. endif
9. endfor
10. RETURN dm && dm = Alphabetic
```

รูปที่ 11
แฟ้มข้อมูล
ที่จะทดสอบโจทย์

Record #	CODE	DETAIL
1	21425	JSX JAPAN
2	53142	@ MADE IN USA
3	43527	532 HAND MADE
4	36241	%50 HONGKONG
5	63518	THAILAND
6	23457	english

```
set logerrors on  * เออเรอจากการคอมพายล์ให้เก็บไว้ในไฟล์ ALPHA.ERR
set notify on     * เมื่อคอมพายล์เสร็จให้แสดงเออเรอบนจอภาพด้วย
compile alpha     * คอมพายล์ต้องไม่มีเออเรอเกิดขึ้น จึงทำต่อไปได้
set procedure to alpha
set talk off
use udf__tst
index on alpha(detail) to udf__tst compact
```

รูปที่ 12 ขั้นตอนในการพิสูจน์โจทย์

รูปที่ 13
ข้อมูลที่ได้จากการ
อินเด็กซ์ควีนฟังก์ชัน
ALPHA

Record #	CODE	DETAIL
6	23457	english
3	43527	532 HAND MADE
4	36241	%50 HONGKONG
1	21425	JSX JAPAN
2	53142	@ MADE IN USA
5	63518	THAILAND

ALPHA แล้ว จะได้การเรียงข้อมูลดังรูปที่ 13

โปรดสังเกตชาแรกเตอร์ที่ขีดเส้นใต้คือความต้องการของโจทย์

หมายเหตุ ก่อนอินเด็กซ์ด้วย UDFs ควรจะใช้คำสั่ง goto top ก่อน เพราะว่าหากเรคอร์ดพอยน์เตอร์ไปชี้ที่ eof( ) จะทำให้ไม่มีข้อมูลที่จะผ่านค่าไปยัง UDFs

BROWSE

คำสั่ง browse เป็นคำสั่งที่ง่ายเหลือเกิน ทำไมต้องเอามาเขียนด้วยก็ไม่รู้ สำหรับฟอกซ์โปรแล้ว คำสั่งนี้ไม่ง่ายครับ ถือว่ายากทีเดียว ไม่เชื่อลองดูรูปแบบคำสั่ง browse ดังในรูปที่ 14

เจอรูปแบบคำสั่งแล้วอย่าว่าแต่ไม่อยากใช้เลยครับ แค่มองก็เวียนหัวซะแล้ว แต่ในความหลากหลายนี้ถูกอัดแน่นไปด้วยคุณภาพจริง ๆ ลองใช้ดูซิครับแล้วจะติดใจ

อนุประโยค (Clauses)

FIELD <field list>

รายชื่อ field ที่ต้องการแก้ไข หากมีมากกว่าหนึ่งต้องคั่นด้วยเครื่องหมาย , และแต่ละรายชื่อ field ยังสามารถระบุออปชันตามหลังได้อีก ซึ่งออปชันที่ตามหลังนี้จะมีผลเฉพาะ field นั้นเท่านั้น

:R ไม่อนุญาตให้มีการเปลี่ยนแปลงข้อมูล

:<col width> กำหนดความกว้างของ field ที่จะรับ/แก้ไขข้อมูล

:V = <expr1> อนุญาตให้มี valid หากฟังก์ชันที่ valid ไปรีเทอร์นค่ากลับมาเป็น .T. การรับ/แก้ไขข้อมูล

```
BROWSE
  [FIELDS<field list>]
  [FOR <expL 1>]
  [FORMAT] [FREEZE <field>]
  [KEY <expr1>[,<expr2>]]
  [LAST]
  [LEDIT] [REDIT]
  [LOCK <expN1>]
  [LPARTITION]
  [NOAPPEND] [NOCLEAR] [NODELETE]
  [NOEDIT|NOMODIFY]
  [NOLGRID] [NORGRID]
  [NOLINK] [NOMENU]
  [NOOPTIMIZE]
  [NORMAL]
  [NOWAIT]
  [PARTITION <expN2>]
  [PREFERENCE <expC1>]
  [SAVE]
  [TIMEOUT <expN3>]
  [TITLE <expC2>]
  [VALID [:F] <expL2> [ERROR <expC3>]]
  [WHEN <expL3>] [WIDTH <expN4>]
  [[WINDOW <window name1>]
  [IN [WINDOW] <window name2> | IN SCREEN]]
  [COLOR [<color pari list> | COLOR SCHEME<expN5>]
```

รูปที่ 14 รูปแบบคำสั่ง browse

จะเลื่อนไป field อื่นถัดไป ถ้ารีเทอร์นค่ากลับเป็น .F. การรับ/แก้ไขข้อมูลจะอยู่ที่เดิม

:F การ valid ด้วย :V จะกระทำเมื่อข้อมูลมีการเปลี่ยนแปลงเท่านั้น หากต้องการให้ valid ทุกครั้งที่ออกจาก field แม้จะไม่มีการเปลี่ยนแปลงข้อมูลก็ตามให้ใช้ :F ร่วมด้วย

:E = <expC1> แสดงเมสเสจ <expC1> เมื่อฟังก์ชันที่ valid ไปรีเทอร์นค่ากลับเป็นมา .F.

:P = <expC2> กำหนดรูปแบบการรับ/แก้ไขข้อมูล โดยใช้ Function code และ Picture code ได้เหมือนที่ใช้กับคำสั่ง SAY/GET ยกเว้นฟังก์ชัน 'M' ตัวเดียวที่ใช้ไม่ได้

:B = <expr2>, <expr3> กำหนดช่วงข้อมูลที่จะรับให้มีค่าระหว่าง <expr2> กับ <expr3> ต้องระบุ :F ด้วย หากต้องการให้ตรวจสอบช่วงข้อมูลในกรณีที่ไม่มีการเปลี่ยนแปลงข้อมูล

:H = <expC3> ใช้ <expC3> แทนชื่อ field

:W = <expL1> จะรับ/แก้ไข field นี้ หาก <expL1> เป็น .T. และจะข้ามไป field ถัดไปถ้า <expL1> เป็น .F.

KEY <expr1>[,<expr2>]

กำหนดช่วงข้อมูลที่จะป้อน โดยช่วงข้อมูล <expr1> และ <expr2> จะมองจากอินเด็กซ์คีย์ หมายความว่าถ้าจะใช้อนุประโยคนี้จะต้องเปิดอินเด็กซ์ไฟล์ด้วย

LAST

แสดงหน้าจอการ browse เหมือนครั้งหลังสุด

[LEDIT] [REDIT]

LEDIT จะแสดงหน้าจอเหมือนการ browse ธรรมดา ส่วน REDIT จะแสดงหน้าจอเหมือนการใช้คำสั่ง CHANGE ปกติแล้วสองอนุประโยคนี้จะใช้ร่วมกับอนุประโยค PARTITION <expN2> เพื่อแบ่งหน้าจอออกเป็นสองส่วน โดย <expN2> เป็นตำแหน่งคอลัมน์ที่จะแบ่งหน้าจอ ซึ่งหน้าจอทั้งสองส่วนนี้จะแสดงข้อมูลในเรคอร์ดที่ตรงกันเสมอ โดยเรคอร์ดที่แสดงอยู่ทางซ้ายจะใช้อ้างอิง แก้ไขไม่ได้ ส่วนเรคอร์ดอยู่ทางขวามือแก้ไขได้เป็นปกติ ถ้าใช้ร่วมกับ LEDIT หน้าจอทางซ้ายจะแสดงเหมือนคำสั่ง CHANGE หน้าจอทางขวาจะแสดงเหมือนการ browse หากใช้ร่วมกับ REDIT จะตรงกันข้าม เช่น

```
browse partition 30 ledit
```

LPARTITION

การใช้อนุประโยค PARTITION ที่ผ่านมา การแก้ไขข้อมูลจะกระทำในส่วนด้านขวาของจอภาพที่ถูกแบ่งแล้ว แต่ถ้ากำหนดอนุประโยค LPARTITION ร่วมด้วย การแก้ไขข้อมูลจะเปลี่ยนมากระทำทางด้านซ้ายมือของจอภาพแทน เช่น

```
browse partition 30 redit lpartition
```

NOAPPEND

ไม่อนุญาตให้เพิ่มเรคอร์ดใหม่ด้วยการกดคีย์ Ctrl+N

NODELETE

ไม่อนุญาตให้ลบข้อมูลด้วยการกดคีย์ Ctrl+T

NOEDIT | NOMIDIFY

ไม่อนุญาตให้มีการแก้ไขข้อมูลใด ๆ

[NOLGRID] [NORGRID]

ไม่แสดงเส้นกริดจอภาพด้านซ้ายหรือด้านขวา

NOLINK

เมื่อข้อมูลทางซ้ายหรือทางขวาเลื่อนไปยังเรคอร์ดอื่น ข้อมูลด้านที่เหลือจะไม่เลื่อนตาม

NORMAL

เมื่อใช้คำสั่ง BROWSE ในขณะที่มีวินโดวส์ใด ๆ activate อยู่ การ browse จะกระทำในวินโดวส์นั้น หากไม่ต้องการต้องระบุอนุประโยค NORMAL ร่วมด้วย

TIMEOUT <expN3>

หากไม่มีการกดคีย์ใด ๆ ภายในระยะเวลา <expN3> (วินาที) จะจบการทำงานคำสั่ง BROWSE ทันที อนุประโยคนี้จะเป็นผลเฉพาะในโปรแกรมเท่านั้น หากใช้ที่คอมมานด์วินโดวส์จะไม่เป็นผล

TITLE <expC2>

แสดง Title ด้วย <expC2> หากไม่ระบุค่าดีฟอลต์จะเป็นชื่อแฟ้มข้อมูลที่ browse ขณะนั้น

VALID [:F] <expL2> [ERROR <expC3>]

คล้ายกับอนุประโยค VALID ของคำสั่ง SAY/GET แต่ค่าที่รีเทอร์นกลับมีได้ 3 ค่า คือ .T., .F., 0 หากฟังก์ชันที่ valid ไปรีเทอร์นค่ากลับมาเป็น .F. การรับ/แก้ไขข้อมูลจะอยู่ที่เดิม พร้อมแสดงเออเรอตาม ค่าที่กำหนดใน <expC3> ถ้ารีเทอร์นค่ากลับเป็น 0 การรอรับ/แก้ไขข้อมูลจะอยู่ที่เดิมเช่นเดียวกัน แต่จะไม่แสดงข้อความเออเรอ หากระบุ :F ตามด้วยจะให้ผลเหมือนที่กล่าวผ่านมา

WHEN <expL3>

จะอนุญาตให้รับ/แก้ไขเรคอร์ดนั้นได้หาก <expL3> เป็น .T.

WIDTH <expN4>

กำหนดความกว้างของ field เท่ากับ <expN4> เท่ากันทุก ๆ field

ตัวอย่างการใช้คำสั่ง BROWSE (ธรรมดา ๆ) ที่คอมมานด์วินโดวส์ร่วมกับแฟ้มข้อมูล brw__tst.dbf มีโครงสร้างและข้อมูลดังในรูปที่ 15 และ 16 และรูปแบบคำสั่งดังรูปที่ 17

คำสั่ง Browse หลังสุดนี้จะรับ

Field	Field Name	Type	Width	
1	ID	Character	4	รหัสประจำตัว
2	FNAME	Character	10	ชื่อ
3	LNAME	Character	10	สกุล
4	BIRTH	Date	8	วันเกิด
5	AGE	Numeric	2	อายุ
6	SALARY	Numeric	6	เงินเดือน
7	STEP	Numeric	4	ขั้นเงินเดือน

รูปที่ 15 โครงสร้างของการใช้คำสั่ง browse ที่คอมมานด์วินโดวส์ร่วมกับแฟ้มข้อมูล brw__tst.dbf

Record #	ID	FNAME	LNAME	BIRTH	AGE	SALARY	STEP
1	2547	SOMPORN	JAIDEE	02/14/60	33	8190	550
2	4379	CHATCHAI	WATTANA	11/08/64	29	7690	500
3	5866	VINAI	NUALCHAN	08/21/68	25	6160	330
4	1072	MOOKDA	SARISO	04/25/70	23	5260	270

รูปที่ 16 ข้อมูลของการใช้คำสั่ง browse ที่คอมมานด์วินโดวส์ร่วมกับแฟ้มข้อมูล brw__tst.dbf

```
use brw__tst
browse field id,fname,lname,birth,salary partition 30 ledit
browse field id,fname,lname partition 40 ledit lpartition
browse field id :h='Identify' :r,fname :h='First name';
      ,lname:w=lempty(fname),salary :r
```

รูปที่ 17 คำสั่ง browse ที่คอมมานด์วินโดวส์ร่วมกับแฟ้มข้อมูล brw__ tst.dbf

Field	Field Name	Type	Width	
1	SALARY	Numeric	6	ตารางเงินเดือน
2	STEP	Numeric	4	ขั้นเงินเดือน

รูปที่ 18 โครงสร้างของการใช้คำสั่ง browse ขั้นแอดวานซ์

Record #	SALARY	STEP
1	4160	200
2	4360	210
3	4570	210
4	4780	230
5	5010	250
6	5260	270
7	5530	300
8	5830	330
9	6160	330
10	6490	350
11	6840	400
12	7240	450
13	7690	500
14	8190	550

รูปที่ 19 ข้อมูลของการใช้คำสั่ง browse ขั้นแอดวานซ์

lname เมื่อ fname ไม่เป็นช่องว่าง ส่วน id และ salary ไม่อนุญาตให้เปลี่ยนแปลงข้อมูล

ตัวอย่างการใช้คำสั่ง BROWSE ขั้นแอดวานซ์ (รันโปรแกรม bw.prg) จะใช้แฟ้มข้อมูล sal__tb.dbf ร่วมด้วย อีกหนึ่งแฟ้ม มีโครงสร้างและข้อมูลดังในรูปที่ 18 และ 19 และมีโปรแกรมของคำสั่ง browse ดังในรูปที่ 20

```
1. ***************
2. *   BW.PRG    *
3. ***************
4. set talk off
6.   clear
7.    set safety off
8.     select 1
9.       use brw__tst
10.      index on id to brw__tst compact
11.         select 2
12.         use sal__tb
13.         index on salary to sal__tb compact
14.         select 1
15.         brow field id :P='9999 :F:V=same__id(id) :E='this ID
            already exist!';
16.         ,fname :H='First name';
17.         ,lname :H='Last name' :W=lempty(fname);
18.         ,birth :F:V=cal__age(birth) :H=Birthday;
19.         ,age :R;
20.         ,salary :F:V=check(salary) :E='this salary out of range';
21.         ,step :R;
22.         when !delete( )
23. set safety on
24. set talk on
25. FUNCTION same__id
26. PARAMETER pid
27.   old__rec = recno( )
28.    if seek(pid)
29.      skip
30.      return!(id = pid)
31.    endif
32.      goto old__rec
33. RETURN .t.
34. FUNCTION check
35. PARAMETER psal
36.  if seek(psal,2)
37.    replace step with sal__tb.step
38.    return .t.
39.  endif
40. RETURN .f.
41. FUNCTION cal__age
42. PARAMETER pb
43.  replace age with year(date( ))-year(pb)
44. RETURN.T.
```

รูปที่ 20 โปรแกรมของคำสั่ง browse

Control Key Shortcuts	
Key	**Action**
Ctrl + A	Select All
Ctrl + C	Copy
Ctrl + D	Do
Ctrl + E	Replace And Find Again
Ctrl + F	Find
Ctrl + G	Find Again
Ctrl + K	Continue
Ctrl + M	Resume
Ctrl + Q	Same as Escape
Ctrl + R	Redo
Ctrl + U	Undo
Ctrl + V	Paste
Ctrl + W	Exit current edit and save any modifications
Ctrl + X	Cut
Ctrl + F1	Cycle windows
Ctrl + F2	Display Command window
Ctrl + F7	Move window
Ctrl + F8	Size window
Ctrl + F9	Minimize frontmost window
Ctrl + F10	Zoom wndow
Escape	Exit current edit without saving modifications; exit dialog without taking action

รูปที่ 21
คีย์ที่ใช้ทั่วไป

ทำความเข้าใจโปรแกรมอีกนิด

บรรทัดที่ 15 : รับ ID เฉพาะตัวเลข หากป้อนตรงกับข้อมูลที่มีอยู่แล้ว จะแจ้งเมสเสจ "this ID already exist!"

บรรทัดที่ 17 : จะรับ LNAME หาก FNAME ไม่เป็นช่องว่าง

บรรทัดที่ 18 : รับ BIRTH พร้อมคำนวณ AGE ด้วย

บรรทัดที่ 19 : AGE ไม่อนุญาตให้แก้ไข

บรรทัดที่ 20 : รับ SALARY โดยข้อมูลที่จะป้อนต้องอยู่ในตารางแฟ้มข้อมูล SAL_TB.DBF เท่านั้น หากป้อนผิดจะแสดงเมสเสจ "this

Control Movement Keys	
Key	**Action**
Right Arrow	Moves cursor one character to the right.
Left Arrow	Moves cursor one character to the left.
Up Arrow	Moves cursor up one line.
Down Arrow	Moves cursor down one line.
PgUp	Moves cursor up one window of text.
PgDn	Moves cursor down one window of text.
Home	Moves cursor to beginning of current line.
End	Moves cursor to end of line.
Ctrl + Right + Arrow	Moves cursor one word to the right.
Ctrl + Left + Arrow	Moves cursor one word to the left.
Ctrl + Home	Moves cursor to begining of text.
Ctrl + End	Moves cursor to end of text.

รูปที่ 22 คีย์เคลื่อนย้ายเคอร์เซอร์

Key Combinations for Selecting Text	
Key	**Action**
Shift + Arrow (Left/Right)	Selects one character at a time.
Shift + Arrow (Up/Down)	Selects one line at a time.
Shift + Ctrl + Arrow (Left + Right)	Selects from cursor to beginning/ end of word.
Shift + Ctrl + End	Selects from cursor to end of text.
Shift + Ctrl + Home	Selects from cursor to beginning of text.
Ctrl + A	Selects all text.

รูปที่ 23 คีย์เลือกกลุ่มเท็กซ์

ตารางที่ 3 Startup switches

STARTUP SWITCHES	
SWITCH	**EFFECT**
–C <PATHNAME>\ <FILE>	ระบุไดรฟ์และ configuration file หากไม่มีชื่อไฟล์ก็จะหมายถึง CONFIG.FP
–E	ไม่ใช้ expanded memory เหมือนกับการกำหนด EMS=OFF ในแฟ้ม CONFIG.FP
–K	สำหรับคอมพิวเตอร์ที่ไม่มีคีย์ F11,F12 เหมือนกับการกำหนด F11,F12=OFF ในแฟ้ม CONFIG.FP
–T	ไม่แสดงโลโกฟอกซ์โปรเมื่อเข้าสู่ระบบ

salary out of range"

บรรทัดที่ 21 : STEP ไม่อนุญาตให้แก้ไข

บรรทัดที่ 22 : เรคอร์ดใดถูก delete ไว้ จะไม่อนุญาตให้เปลี่ยนแปลงข้อมูล

Text editor

ครั้งแรกที่ผู้เขียนใช้ฟอกซ์โปรได้เข้าไปทดสอบเอดิเตอร์ดู ลองกดคีย์ต่าง ๆ ที่ใช้บนเอดิเตอร์ทั่ว ๆ ไป รู้สึกผิดหวังมาก ctrl+y ก็ไม่รับ บล็อกข้อความก็ไม่ได้ อะไร ๆ ก็ไม่มี ให้มาทำไม แล้วใครจะไปใช้ รำพึงรำพันอยู่ในใจ "ทำไม ๆ" ไม่นานนักมีผู้รู้มาแนะให้ไปดูการใช้เอดิเตอร์ในหนังสือ "FoxPro Interface Guide" คราวนี้แหละ qedit ก็ qedit เถอะ เดี๋ยวนี้นาน ๆ ทีผู้เขียนจะไปใช้เทอร์โบปาสคาลบ้าง เทอร์โบซีบ้าง รู้สึกว่าเอดิเตอร์ที่เคยชื่นชอบนักหนาตอนนี้จะติดขัดเสียแล้ว จึงอยากจะแนะนำให้ท่านผู้อ่านลองใช้ดู เริ่มต้นอาจจะอึดอัดอยู่บ้าง ซักพักจะดีขึ้นเรื่อย ๆ ครับ จึงได้เอาตารางคีย์ต่าง ๆ ที่ใช้มาให้ดูดังในรูปที่ 21, 22 และ 23

STARTUP SWITCHES

การรันฟอกซ์โปรเราสามารถกำหนด startup switches ต่อท้ายเข้าไปได้ ซึ่งมีอยู่ 4 ตัวดังตารางที่ 3

ตัวอย่างการเรียกโปรแกรมพร้อมส่ง startup switch โดยใช้คำสั่ง

```
FOXPROL –CC:\NEWPRO.CFG–T
```

หมายถึงเรียกใช้ FOXPRO/LAN มี configuration file ชื่อ NEWPRO.CFG เก็บไว้ใน directory C:\ ขณะกำลังเซตค่า และโหลดแฟ้มต่าง ๆ เพื่อเข้าระบบ จะไม่แสดงโลโกฟอกซ์โปร

เรื่องสั้นทั้ง 6 ที่เล่ามาคงจะได้ถูกนำไปใช้บ้างสำหรับท่านที่ไม่เคยใช้ ท่านที่ชำนาญอยู่แล้วก็ยกไว้เหนือขอบเขตที่จะกล่าวถึง *สวัสดีครับ*